

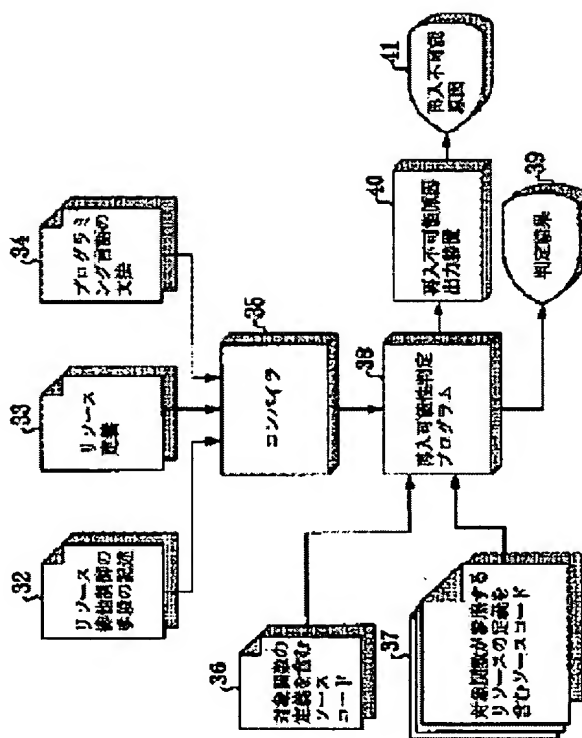
METHOD FOR DECIDING REINPUT POSSIBILITY OF FUNCTION

Patent number: JP2001134431
Publication date: 2001-05-18
Inventor: OMORI MASANORI
Applicant: MITSUBISHI ELECTRIC CORP
Classification:
- international: G06F9/06; G06F11/28
- european:
Application number: JP19990318181 19991109
Priority number(s): JP19990318181 19991109

Report a data error here

Abstract of JP2001134431

PROBLEM TO BE SOLVED: To improve a processing speed and decision reliability by automatically deciding the reinput possibility of a function. **SOLUTION:** This device is provided with a compiler 35 for generating a reinput possibility decision program 38 from a grammar 34 of programming language, resource definition 33, and description 32 of a resource exclusion controlling means, and a reinput possibility decision program 38, and a reinput impossibility cause outputting device 40.



Data supplied from the esp@cenet database - Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-134431

(P2001-134431A)

(43) 公開日 平成13年5月18日 (2001.5.18)

(51) Int. Cl. ⁷	識別記号	F I	テーム(参考)
G 0 6 F	9/06	5 4 0	5 4 0 T 5 B 0 3 3
	9/40	3 1 0	3 1 0 A 5 B 0 4 2
	9/45		3 4 0 F 5 B 0 7 6
	9/46	3 4 0	E 5 B 0 8 1
	11/28		3 2 2 C 5 B 0 9 8
			9/44

審査請求 未請求 請求項の数4 O L (全 9 頁)

(21) 出願番号 特願平11-318181

(22) 出願日 平成11年11月9日 (1999.11.9)

(71) 出願人 000006013

三菱電機株式会社

東京都千代田区丸の内二丁目2番3号

(72) 発明者 大森 正則

東京都千代田区丸の内二丁目2番3号 三

菱電機株式会社内

(74) 代理人 100102439

弁理士 宮田 金雄 (外2名)

Pターム(参考) 5B033 EAG3

5B042 HH10

5B076 EC03

5B081 CC11 CC41

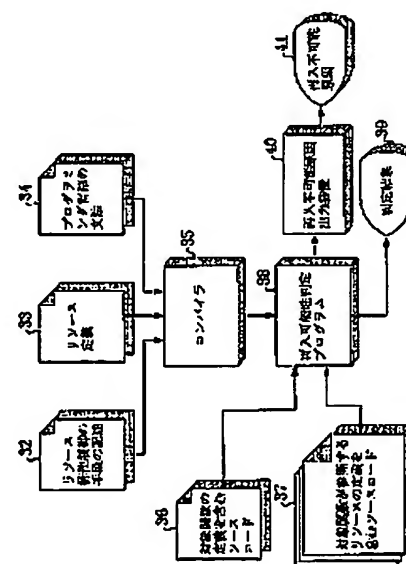
5B098 GA02 GA04 GA08 GD15

(54) 【発明の名称】 関数の再入可能性判定方法

(57) 【要約】

【課題】 関数の再入可能性を自動的に判定し、処理速度と、判定の信頼性を向上させる。

【解決手段】 プログラミング言語の文法34と、リソース定義35、リソース排他制御の手順の記述34から、再入可能性判定プログラム38を生成するコンパイラ35と、再入可能性判定プログラム38と再入不可能原因出力装置40を設けた。



(2)

特開2001-134431

1

【特許請求の範囲】

【請求項1】 時分割または優先度プリエンティブタスクまたは、プロセススケジューリング機能を有するオペレーティングシステムと、関数型プログラミング言語と、前記オペレーティングシステムが提供するリソースの排他制御機能、または、前記オペレーティングシステムに対して提供されたリソースの排他制御機能を持つライブラリを用いることを条件として制作するプログラムに包含される前記プログラミング言語が規定する関数の再入可能性を、前記プログラミング言語の文法と、リソース排他制御の手段の記述と、リソース定義から自動的に判定することを特徴とする関数の再入可能性判定方法。

【請求項2】 前記プログラミング言語が規定する関数の再入可能性を判定することに加えて、判定結果が不可であった場合に判定結果が不可である原因を出力することを特徴とする請求項1記載の関数の再入可能性判定方法。

【請求項3】 前記時分割または、優先度プリエンティブタスクまたは、プロセススケジューリング機能を有するオペレーティングシステムがリソースの排他制御機能を提供しない、または、前記オペレーティングシステムに対して、リソースの排他制御機能を提供するライブラリが存在しないことを条件として制作するプログラムに包含される前記プログラミング言語が規定する関数の再入可能性を判定する請求項1記載の関数の再入可能性判定方法。

【請求項4】 前記時分割または、優先度プリエンティブタスクまたは、プロセススケジューリング機能を有するオペレーティングシステムがリソースの排他制御機能を提供しない、または、前記オペレーティングシステムに対して、リソースの排他制御機能を提供するライブラリが存在しないことを条件として制作するプログラムに包含される前記プログラミング言語が規定する関数の再入可能性を判定することに加えて、判定結果が不可であった場合に判定結果が不可である原因を出力することを特徴とする請求項1記載の関数の再入可能性判定方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】この発明は、プログラムの信頼性を高めるために利用される、関数型プログラミング言語が規定する関数の再入可能性を判定する方法に関する。

【0002】

【従来の技術】時分割または優先度プリエンティブタスクまたは、プロセススケジューリング機能及びそれに類する機能を有するオペレーティングシステムと関数型プログラミング言語を用いることを条件として制作されるプログラムにおいては、複数のタスクまたは、プロセ

2

スからある一定の時間間隔の中で同時に呼び出される関数は再入可能である必要がある。その理由は、関数の動作が予測不能となるためである。従来、関数の再入可能性の判定は人間が手動で行っていたため、判定に時間がかかり、また、判定の信頼性が十分ではなかった。

【0003】

【発明が解決しようとする課題】時分割または優先度プリエンティブタスクまたは、プロセススケジューリング機能を有するオペレーティングシステムと関数型プログラミング言語を用いることを条件として制作されるプログラムにおいて、関数の再入可能性は定義できる。ある関数を複数のタスクまたはプロセスから、その関数仕様を逸脱する動作をすることなしに同時に呼び出すことができれば、その関数は再入可能であるという、関数仕様を逸脱する動作は、グローバルリソース（複数のタスクコンテキストで共有されるリソース）へのアクセスが適切に排他制御されていない場合に起こる。関数が再入可能ではない場合、タスクまたはプロセスの動作が不安定になる可能性が高く、不具合が生じた場合に、不具合の修正作業が極めて困難になるという特徴がある。手動による判定では判定を行おうとする個人の技術によって判定の精度が異なる可能性があり、また、完全な判定である保証はない。さらに、手動による判定では判定に時間がかかるという問題がある。この発明の目的は、前記問題点を解決するために自動で関数の再入可能性の判定を行うことにある。

【0004】

【課題を解決するための手段】この発明においては、判定を行う対象である関数が定義されたソースコードが使用しているプログラミング言語の構文とリソースの排他制御を行う手段の記述と、判定を行う対象である関数が定義されたソースコードが使用しているプログラミング言語の構文とリソースの排他制御を行う手段の記述と入力として、関数の再入可能性を自動判定するプログラムを生成するコンパイラを用いて、関数の再入可能性を自動判定するプログラムを生成し、生成されたプログラムの入力を、判定しようとする関数の定義を含むソースコード及び、対象関数からアクセスされるリソースの定義を含むソースコードとし、判定結果を出力するための装置を出力として、生成されたプログラムを動作させることによって、前記課題を解決する。

【0005】さらに詳細に記述すると、前記関数の再入可能性を自動判定するプログラムを生成するコンパイラによって生成された、関数の再入可能性を自動判定するプログラムは、判定を行う対象である関数が定義されたソースコードが使用しているプログラミング言語の構文にしたがって、入力されたソースコードを構文解析し、再入可能性判定を行う対象である関数内でアクセスされるすべてのリソースに対して、スコープを決定し、再入可能性判定を行う対象である関数内でアクセスされるリ

(3)

特開2001-134431

3

ソースのスコープの種類に応じて判定の手段を決定し、前記判定の手段を用いて判定を行う。再入可能性判定を行う対象である関数内でアクセスされるすべてのリソースに対して、判定が正であれば、当該関数は再入可能であると判定される。

【0006】

【発明の実施の形態】実施の形態1。図1はこの発明のハードウェア構成を示すブロック図、図2はこの発明の実施の形態1におけるソフトウェア構成を示すブロック図、図3は開数の再入可能性判定の処理手順を示すフローチャート、図4はリソースの排他制御の手段を表す模式図である。

【0007】図2に示す、この発明の実施の形態1を示すソフトウェア構成において、再入可能性判定プログラムを出力するコンパイラ11は、図1に示すディスク装置5に格納されており、入力装置7から起動されることにより、メモリ3にロードされて中央演算処理装置2において動作を開始し、同じく入力装置7から指定された、ディスク装置5にファイルとして格納されているプログラミング言語の文法10と、リソース定義9と、リソース排他制御の手段の記述8をI/O制御装置4を介してメモリ3上に読み出し、これを入力として、再入可能性判定プログラム14をメモリ3上に生成し、起動する。

【0008】さらに、再入可能性判定プログラム14は、中央演算処理装置2において動作し、判定しようとする関数の定義を含むソースコード12及び、対象関数が参照するリソースの定義を含むソースコード13をディスク装置5から入力し、再入可能性判定結果15を、出力装置6にI/O制御装置4を介して出力する。図3のフローチャートは、この発明の実施の形態1における再入可能性を判定するプログラム14の動作を示している。実施の形態1における再入可能性を判定するプログラムは、まず、処理手順16において、対象関数を解析し、判定対象の関数がアクセスするリソースのリストを作成する（処理手順17）。次に、処理手順18において、対象関数が記述されたプログラミング言語の言語仕様におけるスコープの種類に従って、アクセスリソースリストをローカルリソースリストと、グローバルリソースリストに分割する。アクセスするリソースがすべてローカルリソースであれば、すなわち、グローバルリソースリストが空集合であれば、対象関数が複数のタスク、または、プロセスから呼び出されても競合が発生しないので、対象関数は再入可能である（判定19）。

【0009】グローバルリソースリストが空集合でない場合は、そのリソース一つ一つに対して適切に排他制御がなされているか判定する（判定21）必要がある。対象となるリソースは図4に示す、関数定義26において、排他制御に入る処理27、及び、排他制御から出る処理28に囲まれた排他制御領域29に存在しなければ、適

4

切に排他制御されていない。図4においては、グローバルリソースaへのアクセス30は排他制御されているが、グローバルリソースbへのアクセス31は排他制御されていない。よって、グローバルリソースaは図4に示す以外にアクセスがなければ、適切に排他制御されているといえる。図3のフローチャートにおいて、処理手順20～24までの処理をグローバルリソースリスト中のすべてのリソースに対して適用すると、OKリスト、NGリストが生成される。ここで、NGリストが空集合であれば、すべてのグローバルリソースへのアクセスは適切に排他制御されているということであるから、対象関数は再入可能であると判定される。NGリストがひとつでも要素を含んでいれば、再入可能ではないと判定される（判定25）。

【0010】実施の形態2。図5に示す、この発明の実施の形態2を示すソフトウェア構成において、コンパイラ35はディスク装置5に格納されており、入力装置7から起動されることにより、メモリ3にロードされ、中央演算処理装置2において動作を開始し、同じく入力装置7から指定された、ディスク装置5にファイルとして格納されている、プログラミング言語の文法34と、リソース定義33と、リソース排他制御の手段の記述32をI/O制御装置4を介してメモリ3上に読み出し、これを入力として、再入可能性判定プログラム38を生成し、起動する。再入可能性判定プログラム38は中央演算処理装置2において動作し、判定しようとする関数の定義を含むソースコード36及び、対象関数が参照するリソースの定義を含むソースコード37を入力として、再入可能性判定結果39と、判定が不可の場合に、再入不可能原因出力装置40を通して、対象関数が再入可能でない原因41をI/O制御装置4を介して、出力装置6に出力する。

【0011】図6のフローチャートにおける、処理手順42～51の処理に関しては実施の形態1の動作を記述した図3のフローチャートと同一である。実施の形態2では、実施の形態1に加えて、NGリストに存在するグローバルリソースへの判定対象関数におけるアクセス箇所を、対象関数が再入可能でない原因41として、出力する（処理手順45）。

【0012】実施の形態3。図7に示す、この発明の実施の形態3を示すソフトウェア構成において、コンパイラ56はディスク装置5に格納されており、入力装置7から起動されることにより、メモリ3にロードされて、中央演算処理装置2において動作を開始し、同じく入力装置7から指定された、ディスク装置5にファイルとして格納されているプログラミング言語の文法55と、リソース定義54を入力として、再入可能性判定プログラム59をメモリ3上に生成し、これを起動する。再入可能性判定プログラム59は中央演算処理装置2において動作し、判定しようとする関数の定義を含むソースコー

(4)

特開2001-134431

5

6

ド57及び、対象関数が参照するリソースの定義を含むソースコード58を入力として、再入可能性判定結果53を出力装置6にI/O制御装置4を介して出力する。

【0013】図8のフローチャートは、この発明の実施の形態3における再入可能性を判定するプログラム59の動作を示している。実施の形態3では、リソースの排他制御の手段が提供されない場合であるから、処理手順63において、アクセスリソースリストをローカルリソースリストと、グローバルリソースリストに分割した時点で、グローバルリソースリストが空集合でなければ、再入可能ではないと決定することができる（判定64）。

【0014】実施の形態4。図9に示す、この発明の実施の形態4を示すソフトウェア構成において、コンパイラ7はディスク装置5に格納されており、入力装置7から起動されることにより、メモリ3にロードされて、中央演算処理装置2において動作を開始し、同じく入力装置7から指定された、ディスク装置5にファイルとして格納されているプログラミング言語の文法66と、リソース定義65を入力として、再入可能性判定プログラム70をメモリ3上に生成し、起動する。再入可能性判定プログラム70は中央演算処理装置2において動作し、判定しようとする関数の定義を含むソースコード68及び、対象関数が参照するリソースの定義を含むソースコード69を入力として、再入可能性判定結果71と、判定が不可の場合に、再入不可能原因出力装置72を適して、対象関数が再入可能でない原因73を出力装置6にI/O制御装置4を介して出力する。

【0015】図10のフローチャートにおける、処理手順74～76に関しては実施の形態3の動作を記述した図8の処理手順61～63と同一である。実施の形態4では、実施の形態3に加えて、グローバルリソースリストに存在するリソースへの対象関数におけるアクセス箇所を、対象関数が再入可能でない原因として出力する（処理手順78）。

【0016】

【発明の効果】第1の発明は、関数の再入可能性判定プログラム14により、判定処理の自動化を行うことができる。また、プログラミング言語の文法10、リソースの定義9、リソース排他制御の手段の記述8から関数の再入可能性を判定するプログラムを生成するコンパイラ11を設けることにより、プログラミング言語、リソース、排他制御の手段のうちひとつまたは複数異なる環境においても容易に判定処理の自動化を提供することができる。

【0017】また、第2の発明は、第1の発明の効果に加えて、対象関数が再入可能でなかった場合に、再入不可能原因を出力する装置40を設けることにより、再入可能ではない原因を特定することができる。

【0018】また、第3の発明は、リソースの排他制御の手段が提供されなかった場合に、関数の再入可能性判定プログラム59により、判定処理の自動化を行うことができ、処理速度の向上と、判定の信頼性の向上を得ることができる。また、プログラミング言語の文法55、リソースの定義54から関数の再入可能性を判定するプログラム59を生成するコンパイラ56を設けることにより、プログラミング言語、リソース、排他制御の手段のうちひとつまたは複数異なる環境においても容易に判定処理の自動化を提供することができる。

【0019】また、第4の発明は、第3の発明の効果に加えて、対象関数が再入可能でなかった場合に、再入不可能原因を出力する装置72を設けることにより、再入可能ではない原因を特定することができる。

【図面の簡単な説明】

【図1】 この発明のハードウェアブロック図である。

【図2】 この発明の実施の形態1を示す関数の再入可能性判定装置のソフトウェアブロック図である。

【図3】 この発明の実施の形態1の処理手順を示すフローチャートである。

【図4】 リソース排他制御の手段を表す模式図である。

【図5】 この発明の実施の形態2を示す関数の再入可能性判定装置のソフトウェアブロック図である。

【図6】 この発明の実施の形態2の処理手順を示すフローチャートである。

【図7】 この発明の実施の形態3を示す、関数の再入可能性判定装置のソフトウェアブロック図である。

【図8】 この発明の実施の形態3の処理手順を示すフローチャートである。

【図9】 この発明の実施の形態4を示す、関数の再入可能性判定装置のソフトウェアブロック図である。

【図10】 この発明の実施の形態4の処理手順を示すフローチャートである。

【符号の説明】

1 計算機、2 中央演算処理装置、3 メモリ、4 I/O制御装置、5 ディスク装置、6 出力装置、7 入力装置、8 実施の形態1におけるリソース排他制御の手段の記述、9 実施の形態1におけるリソース定義、10 実施の形態1におけるプログラミング言語の文法、11 実施の形態1における再入可能性判定プログラムを出力するコンパイラ、12 実施の形態1における対象関数を含むソースコード、13 実施の形態1における対象関数が参照するリソースの定義を含むソースコード、14 実施の形態1における再入可能性判定プログラム、15 実施の形態1における判定結果、26 関数定義、29 排他制御領域、32 実施の形態2におけるリソース排他制御の手段の記述、33 実施の形態2におけるリソース定義、34 実施の形態2におけるプログラミング言語の文法、35 実施の形態2に

(5)

特開2001-134431

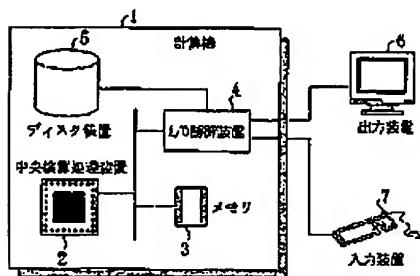
7

8

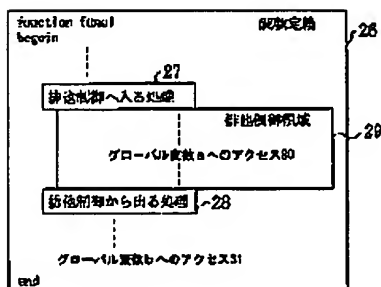
おける再入可能性判定プログラムを出力するコンパイラ、36 実施の形態2における対象関数を含むソースコード、37 実施の形態2における対象関数が参照するリソースの定義を含むソースコード、38 実施の形態2における再入可能性判定プログラム、39 実施の形態2における判定結果、40 実施の形態2における再入不可能原因出力装置、41 実施の形態2における再入不可能原因、54 実施の形態3におけるリソース定義、55 実施の形態3におけるプログラミング言語の文法、56 実施の形態3における再入可能性判定プログラムを出力するコンパイラ、57 実施の形態3における対象関数を含むソースコード、58 実施の形態3における対象関数が参照するリソースの定義を含むソース

* スコード、59 実施の形態3における再入可能性判定プログラム、60 実施の形態3における判定結果、65 実施の形態4におけるリソース定義、66 実施の形態4におけるプログラミング言語の文法、67 実施の形態4における再入可能性判定プログラムを出力するコンパイラ、68 実施の形態4における対象関数を含むソースコード、69 実施の形態4における対象関数が参照するリソースの定義を含むソースコード、70 実施の形態4における再入可能性判定プログラム、71 実施の形態4における判定結果、72 実施の形態4における再入不可能原因出力装置、73 実施の形態4における再入不可能原因。

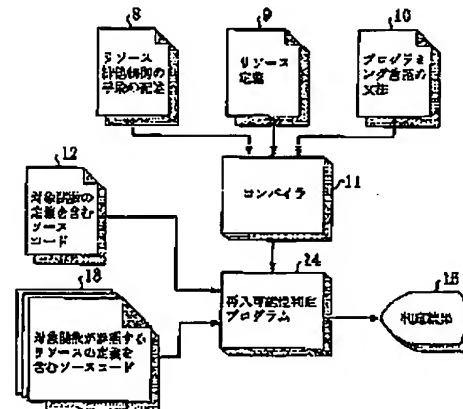
【図1】



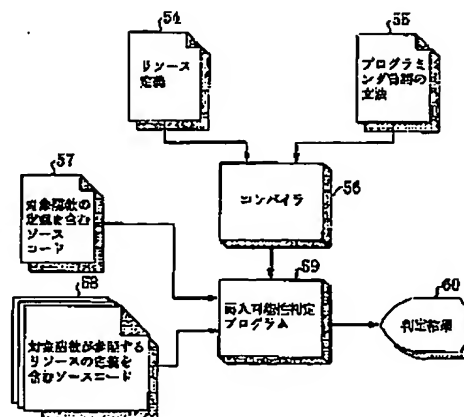
【図4】



【図2】



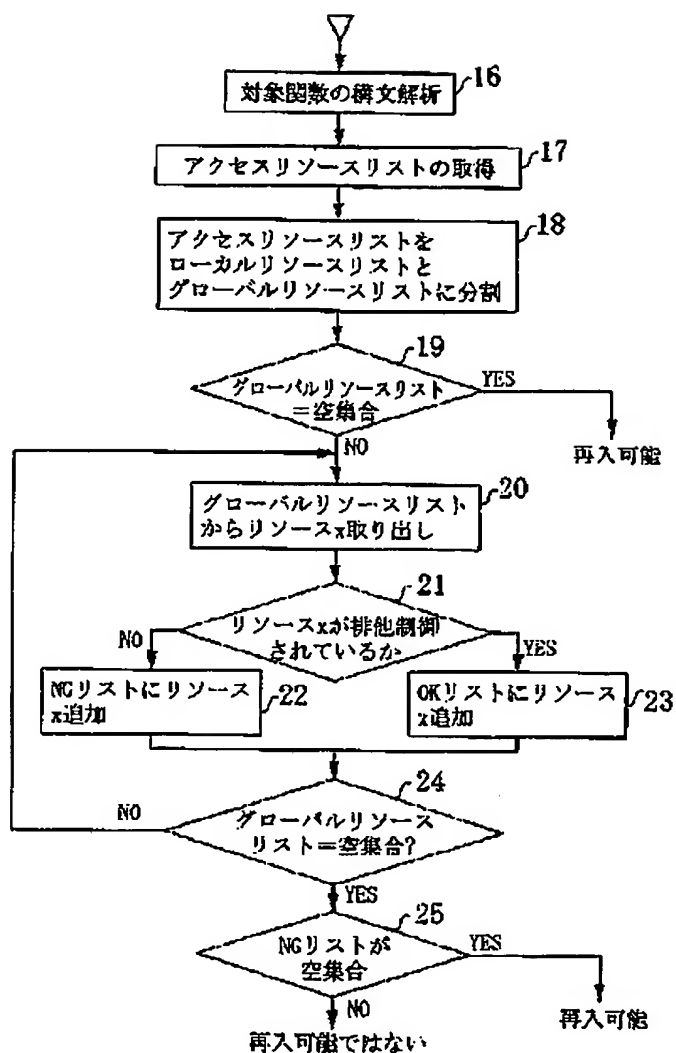
【図7】



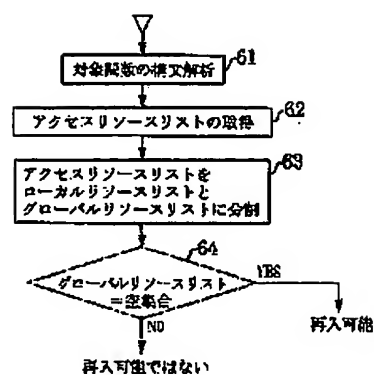
(5)

特開2001-134431

【図3】



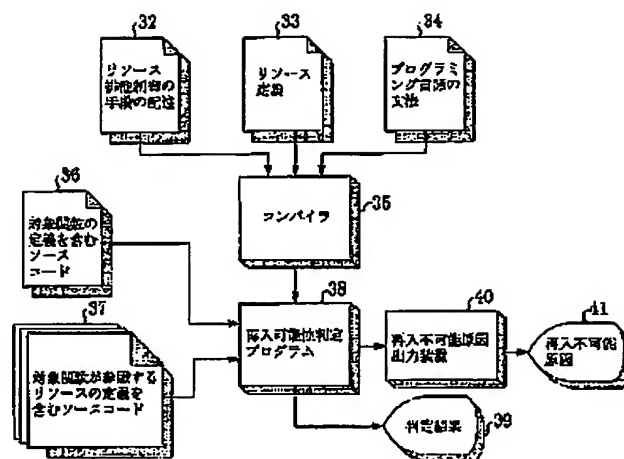
【図8】



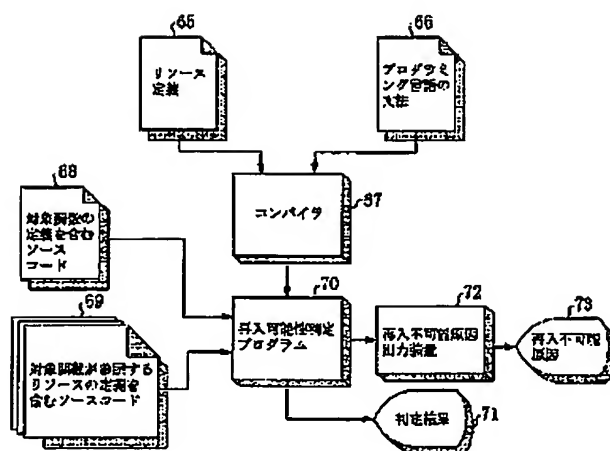
(7)

特開2001-134431

【図5】



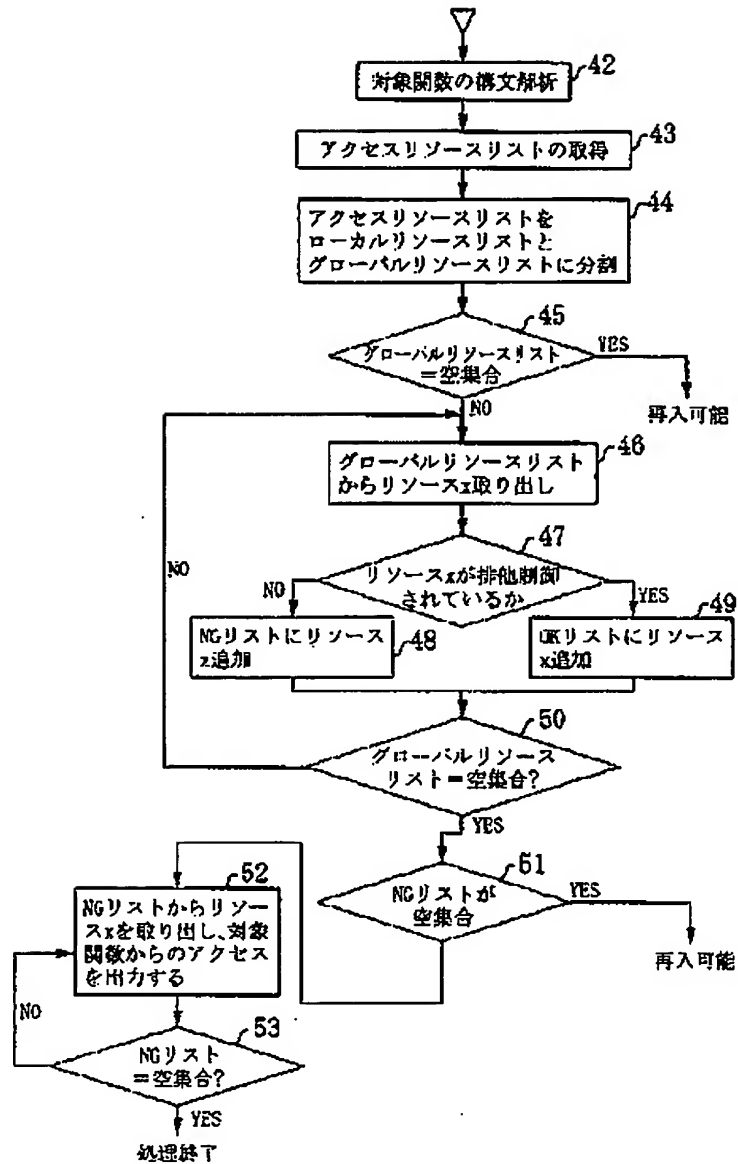
【図9】



(8)

特開2001-134431

【図6】



(9)

特開2001-134431

【図10】

